SRAM-PUF Based Entities Authentication Scheme for Resource-constrained IoT Devices

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2020.3032518, IEEE Internet of Things Journal

SRAM-PUF Based Entities Authentication Scheme for Resource-constrained IoT Devices

Fadi Farha, Huansheng Ning*, *Senior, IEEE,* Karim Ali, Liming Chen, *Senior, IEEE,* and Christopher Nugent

***Abstract*—With the development of the cloud-based Internet of Things (IoT), people and things can request services, access data, or control actuators located thousands of miles away. The entity authentication of the remotely accessed devices is an essential part of the security systems. In this vein, Physical Unclonable Functions (PUFs) are a hot research topic, especially for generating random, stable, and tamper-resistant fingerprints. This paper proposes a lightweight, robust SRAM-PUF based entity authentication scheme to guarantee that the accessed end devices are trustable. The proposed scheme uses Challenge-Response Pairs (CRPs) represented by re-ordered memory addresses as challenges and the corresponding SRAM cells' startup values as responses. The experimental results show that our scheme can efficiently authenticate resources-constrained IoT devices with a low computation overhead and small memory capacity. Furthermore, we analyze the SRAM-PUF by testing the PUF output under different environmental conditions, including temperature and magnetic field, in addition to exploring the effect of writing different values to the SRAM cells on the stability of their startup values.**

***Index Terms*—Entity authentication, SRAM-PUF, Physical security.**

I. INTRODUCTION

THE Internet of Things (IoT) world and its applications are growing rapidly, and so is the number of connected devices [1]. Many people and services access devices and equipment, which can be thousands of miles far away, in terms of remote sensing and controlling. Such a remote access process requires a high degree of trust, which guarantees the security and authenticity of the accessed devices [2].

For IoT devices, especially the low-cost manufactured devices, security has always been a challenge [3]. In real-world applications, various reliable and robust cryptographic algorithms can authenticate the devices' entities, such as the Public Key based algorithms [4] and asymmetric handshake based cryptosystem [5]. However, applying these traditional cryptographic algorithms and conventional security solutions to IoT devices working in the sensing layer faces some hardware-related difficulties. Firstly, most of these devices are low-cost and resource-constrained, which affect their processing power and storage capacity. Hence, they are unable to run the security methods that require intensive computational operations [6]. Secondly, many IoT devices do not have Internet access, so there is no direct connection to the remote authenticator.

Many researchers worked on identity-based cryptosystems and fingerprint schemes, such as Shamir [7], who built his scheme using a private key for proving the device/person identity. Unfortunately, the Public Key Infrastructure (PKI) based scheme requires computing power, which is unavailable in the resource-constrained devices. Recently, there were many proposed schemes, such as identity-Based encryption from lattices [8], attribute-based encryption for circuits [9], certificateless public key authenticated encryption [10], and fully homomorphic encryption [11], which depend on the public keys concept in some stage during the encryption phase. Those algorithms are effective and ensure the identity of the devices but still require intensive computation.

In addition to the computing power, another challenge faces the authentication schemes represented by exposing the secret keys or the fingerprints stored in Nonvolatile Memory (NVM) of the local devices, which can affect the system security severely. The IoT devices are usually vulnerable to various attacks, including physical attacks such as devices tampering, reading out the devices' NVM, and extracting the devices' secrets [12]. These vulnerabilities were of the early motivations for using Physical Unclonable Functions (PUF), which does not require storing the keys on the devices [13]. Besides, tampering the PUF units would damage their structure and make them useless. Hence, in this paper, a lightweight PUF-based entity authentication scheme is proposed. Hence, in this paper, a lightweight PUF-based entity authentication scheme is proposed. This scheme is applicable to the resource-constrained IoT devices, and the fingerprints generated using PUF are robust against physical attacks.

PUF can be defined as a fingerprint of the physical objects [14]. Like human beings' fingerprints, each object has unique variables, features, or behavior that makes it different from other objects of the same type. PUF is a security primitive that exploits the uncontrolled and unavoidable physical variations generated during the fabrication process of the integrated circuits (ICs). Since the variations sometimes cause random, but stable, mismatches, they can be used as fingerprints, private secret keys, or as a root of trust in physical system structures [15].

Using the PUF is considered as a secure alternative to the traditional storage of the secret keys and IDs [14]. Besides, the PUF-based protocols used in authentication and security are lightweight and significantly simplified to reduce the

Huansheng Ning, Fadi Farha, and Karim Ali are with the School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing, China. (e-mail: ninghuansheng@ustb.edu.cn, fadi_farha@xs.ustb.edu.cn, karim_ali_ali@xs.ustb.edu.cn)

Liming Chen and Christopher Nugent are with the School of Computing, Ulster University, Jordanstown, Northern Ireland, UK.(e-mail: l.chen@ulster.ac.uk, cd.nugent@ulster.ac.uk)

Huansheng Ning is the corresponding author.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2020.3032518, IEEE Internet of Things Journal

overhead at the end devices. They provide robust security features with a lower computational cost compared with the PKI-based cryptosystem [16]. These protocols usually use Challenge-Response Pairs (CRPs) as follows: in the enrollment phase, the authenticator challenges the PUF, measures the PUF responses, and stores the CRPs in its database. The challenging mechanism depends on the PUF type. In the authentication phase, the authenticator chooses one of the CRPs, challenges the PUF, and compares the generated response with the recorded one. The PUF responses sometimes are noisy and need correction, so researchers usually use Error Correction Code (ECC) algorithms to fix them by generating helper data [17]. According to the number of CRPs, the PUF can be classified as strong PUF if the number of CRPs is significant. Besides, adding a new component to the PUF unit can increase the CRPs exponentially. The weak PUF, on the other side, has few CRPs, and adding a new component to the PUF unit can increase the CRPs linearly [15].

PUF has been used in the authentication schemes. In [18], Wallrabenstein, J.R. used PUF as a part of the public-private keys scheme. He stored a public challenge and its helper data on the end device to be used later in generating the private key [18]. However, the proposed authentication scheme mainly runs a hashing function on a random nonce and a private key more than using the PUF-based CRPs. Besides, repeating the nonce will break the security of the authentication process (Man-in-the-middle attack). In [19], Braeken, A. used a strong PUF based public-private keys scheme in the authentication process between two end devices, which is done through a server. However, the author did not discuss the size of the helper data used for correcting the noisy PUF responses, which is significant. Besides, the computation overhead is relatively high. Other researchers used strong PUFs based CRPs schemes such as [20] and [21], which usually needs a special design for PUF units, which causes extra cost, and requires a memory capacity to store the significant size of helper data. On the other hand, obfuscated challenge-response protocols [22] need no encryption, and they are robust against machine learning attacks. However, the CRPs should be chosen carefully in the initial phase, and the used PUF needs to be strong.

While early proposed PUF types had their own special designs [15], a recent trend is to get unique features from the already deployed devices without adding new equipment or replacing the existing devices. Some researchers tried to extract a fingerprint from the devices' memories, such as SRAM-PUF [14], DRAM-PUF [23], and flash-PUF [24]. This paper focuses on the SRAM-PUF and how we can overcome its shortages.

Static Random Access Memory (SRAM)-PUF was first introduced almost simultaneously in [25] by Guajardo et al. and [26] by Holcomb et al. The authors in these papers noticed that the SRAM cells startup with a random value of 0 or 1. After reading the Startup Values of the SRAM Cells (SVRCs) multiple times, the authors found out that these SVRCs are relatively stable and unchangeable. Besides, different SRAM chips have different SVRCs indicating that SVRCs can be potential fingerprints for the local devices in which they are installed. The SRAM cell consists of six

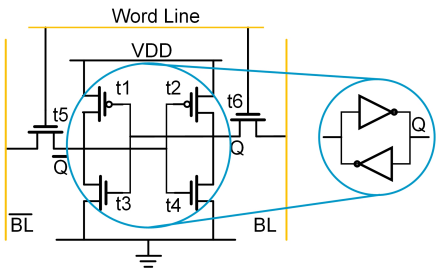


Fig. 1. SRAM cell

transistors; two of them (t_5, t_6) are access transistors, and the other four (t_1, t_2, t_3, t_4) make two cross-coupled inverters as shown in Fig.1. SVRCs are leaded by the mismatches in the CMOS transistors of the SRAM cell that make the cell startup value zero or one. These mismatches are caused by uncontrolled variations on the atom level at the fabrication process. According to the startup values, the memory cells can be classified into 1) skewed cells, which always startup with the same value as zero "0-skewed cell" or as one "1-skewed cell". 2) Neutral cells, which have no strong tendency to any value and start with a random value each time. The skewed cells are suitable to be used as a fingerprint because of their high stability [14].

The startup values of SRAM cells are the entropy source of SRAM-PUF. Since that each cell can only generate 1-bit output, adding a new SRAM cell can only increase the CRPs by 1, and that is why the SRAM-PUF is classified as a weak PUF [15]. For SRAM-PUF to be used in authentication, the cells' addresses are used as a challenge, and the startup values stored in the corresponding cells are the response. However, challenged addresses can only be used once because their content is exposed to the outside world. Therefore, SRAM-PUF cannot be used directly in the authentication process because of the lack of entropy, so in this paper, we add a hashing function to SRAM-PUF output after re-ordering its bits to overcome this shortage.

In addition to the lack of entropy, stability is another issue that needs to be addressed. SRAM-PUF output sometimes is noisy and unstable under different environmental conditions such as high temperature and high voltage. Besides, since the SRAM-PUF consists of electrical components, it is also vulnerable to the aging effect. Some researchers have done some experiments to study the effect of the circuit age on SVRCs stability, whether by running the device under high temperature and using high voltage to accelerate the circuit aging where the rate of unstable cells increased by 5.3% to 7.2% [27] or by reading the SRAM-PUF output continuously for a long time (two years) where the rate of unstable cells increased by 2.49% to 2.97% [28]. The authors in [27] proposed a method called anti-aging to help to mitigate the aging effect by keeping $|V_{th,t1} - V_{th,t2}|$ big enough to ensure the stability of SVRCs. However, in general, some PUF protocols accept the partially unstable PUFs. The others require the PUF output to be %100 stable, especially when the PUF is used to generate secret keys, which require ECC algorithms to fix the PUF output.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2020.3032518, IEEE Internet of Things Journal

In this paper, we work with the SRAM-PUF because the startup values of SRAM show good PUF features, including stability, randomness, and uniqueness. Besides, SRAM-based Field Programmable Gate Arrays (FPGAs) occupy the majority of the market today. Most IoT devices, which work as sensors and actuators, are equipped with SRAM, such as smartphones and FPGAs produced by big companies, including Xilinx and Altera, Arduino, Texas Instruments, and others [25]. As a result, the devices can benefit from the PUF features without installing any new equipment. The key contributions in this paper are as follows:

1) We analyze the startup values of the SRAM-PUF cells and present some significant statistics about them. Besides, we test the SRAM-PUF under different environmental conditions including temperature, magnetic field in addition to showing the effect of using the SRAM-PUF cells by system on their startup values.
2) An SRAM-PUF based entity authentication scheme for the resource-constrained devices has been proposed. Also, the new scheme is tested and explored in detail to fit with the different types of end devices. The proposed scheme enables us to have a large number of CRPs with minimum cost, minimum storage for the helper data, and without changing the structure of SRAM.

The remaining part of the paper is organized as follows: Section II describes the proposed SRAM-PUF authentication scheme phases in detail. In Section III, we have evaluated the scheme by doing some tests on the SRAM cells and providing some statistics and information to be used while implementing the proposed scheme. Results and discussion are presented in section IV, and Section VI concludes the paper.

II. SRAM-PUF BASED AUTHENTICATION SCHEME

Our scheme is meant to authenticate the end devices (provers) by the smart gateway (authenticator). It consists of two phases: the enrollment phase and the authentication phase, as shown later in this section. Firstly we explain what we mean by the smart gateway.

A. Smart Gateway

A simple presentation of the cloud-based IoT structure is shown in Fig.2. In this structure, the sensors collect and transmit data to the cloud through the gateway. The local network components, in such scenarios, are digital devices consisting of communication modules attached to microcontrollers. The gateway must handle different communication protocols to manage the connection between the IoT devices in the sensing layer and the cloud. That is why we use the term “smart gateway.” Considering smart home or smart healthcare devices as an instance, most sensors and actuators use Bluetooth, ZigBee, or Wi-Fi (with private IP addresses) to transfer their data. Therefore, the remote services cannot directly access the end devices to perform entity authentication because they are using incompatible addresses. Instead, the edge computing devices close to the end devices can do this task [29].

In this paper, a raspberry pi 3 is used as a smart gateway. It has Bluetooth, Wi-Fi, LAN and WAN interfaces. Besides,

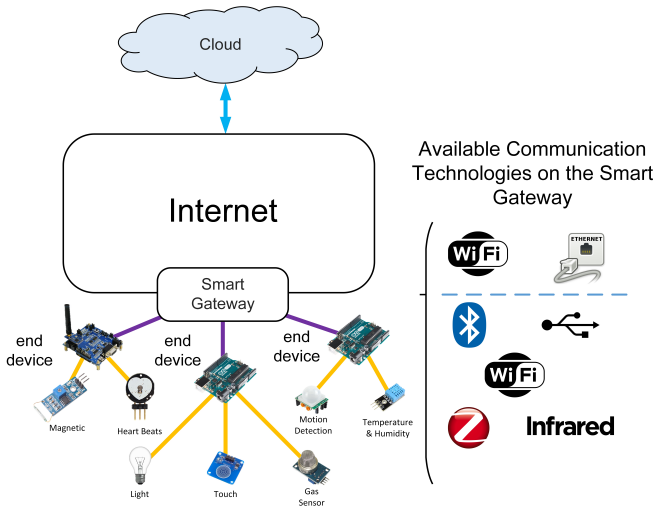


Fig. 2. The System Structure

we attached an XBee module to the smart gateway to transmit/receive ZigBee frames. That is not the only choice. There are also other products available in the market under the name "IoT gateway," which supports most of the standard communications protocols used in IoT. As shown in Fig.2, the smart gateway plays the role of a local authenticator. If any remote device or any service on the cloud needs to access a sensor attached to an end device inside the local network, the smart gateway will be responsible for authenticating the identity of the local end device. The smart gateway is an optimal choice in such a scenario for the following reasons:

1) It is directly connected to end devices and can carry out the authentication process locally. That will keep the secrets of the local network safe from being transmitted over unsecured networks (the Internet), preserve privacy, and protect the local network structure from being exposed to the outside world.
2) It has non-IP compatible interfaces for local communications and an IP-compatible interface for establishing remote connections using robust security algorithms.

B. Enrollment phase

In this phase, a stable fingerprint (100 bits) is extracted from the SRAM of the IoT end device. There is no need to store the fingerprint in the NVM of the end device because the fingerprint can be generated on-time when the entity authentication is required.

Since IoT end devices have programmable microcontrollers or microprocessors, reading the startup values of SRAM requires writing the code in the setup part of the device firmware making memory reading the first instruction to be executed by the microprocessor. That ensures the read values are the startup values of the SRAM cells, not new values written by the running program.

As shown in Fig.3, for generating the stable fingerprint, we read the startup values of a part of the device’s SRAM multiple times - in this paper, startup values of 1KB of SRAM was read for 50 times. Then, a stability map is created to show

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2020.3032518, IEEE Internet of Things Journal

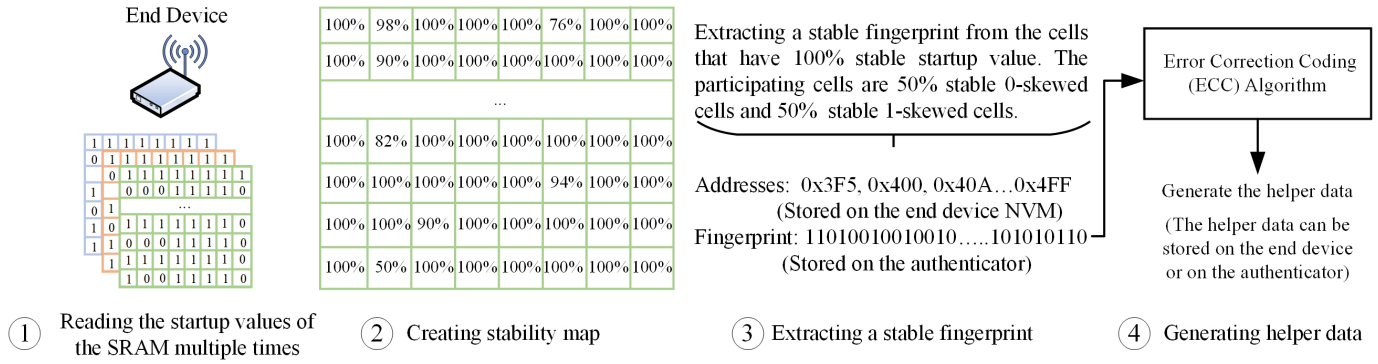


Fig. 3. Enrollment Phase

the stability of SVRCs. After that, a simple algorithm runs to choose the SRAM-PUF participating cells as follows:

1) 50% of the fingerprint are chosen from the 1-skewed cells, and the other 50% are chosen from the 0-skewed cells.
2) The algorithm chooses the participating cells starting with 100% stable cells. If there are not enough 100% stable cells to build the fingerprint, the algorithm will choose from the 99% stable cells, and so on until a full fingerprint is built entirely.

After building the stable fingerprint, it is passed to an ECC algorithm for generating the helper data, as shown in Fig.3 step 4. Helper data, which help correct the generated fingerprint if some bits change for some reason, are stored either on the end device in NVM or on the authenticator.

At the end of this stage, the fingerprints of all end devices are stored in the authenticator device (i.e., smart gateway). Sending the fingerprints from end devices to the smart gateway is the most critical step in this stage. The fingerprints can be inserted manually into the smart gateway or sent over the air to the gateway if the communication protocol used between the end device and the smart gateway is secured, and the packets are encrypted. Exposing the fingerprints in this stage undermines system security and further makes the future authentication requests meaningless.

C. Authentication phase

When an entity authentication is required, the authenticator challenges the end device and measures its response. In our proposed scheme, the authenticator sends the addresses of the stable SRAM cells in a random order. Using this method, the authenticator will have $100 = 9.3 * 10^{157}$ available challenges. After receiving the challenge, the end device does not send the startup values of these addresses directly. Instead, it runs some operations on the PUF value and applies a hashing algorithm to hide the original values of the SRAM cells. Firstly, we list some of the terms used in the authentication phase steps:

- id: the end device ID
- nonce: a random number with a length of 4 bytes
- challenge: the addresses of the SRAM-PUF cells in a random order
- PUF: the SRAM-PUF output with a length of 100 bits
- RPUF: the re-ordered SRAM-PUF output with a length of 100 bits
- E_{NWK}: encryption using a network key
- H: hashing function

As shown in Fig.4, the authentication process consists of some steps. Most of them run on the authenticator and the end device simultaneously as follows:

1) At the beginning of the authentication process, the authenticator generates a challenge and a random nonce. Then, it sends them up with the device ID to the end device. The message is encrypted usinga network (NWK) key related to the communication protocol $E_{NWK}(id, nonce, challenge)$. We consider that the connection between the gateway and the end device is secured using an NWK key regardless of the used communication protocol, whether it is Bluetooth, ZigBee, Wi-Fi, or any other communication method.
2) After receiving the authentication request, the end device decrypts the message and obtains the sent nonce and challenge. Then, the end device generates its PUF output and corrects it using ECC and the helper data. Next, the end device re-orders the PUF output bits according to the received challenge.
3) An XoR operation is carried out between the original PUF output, the re-ordered PUF output, and the nonce. Then, a hashing function is added to the output of the XoR operation $H(nonce \oplus PUF \oplus RPUF)$. Simultaneously, on the authenticator side, the authenticator uses the device ID as an index to search the device's fingerprint (PUF output), which is recorded in its table. After that, the authenticator conducts the same process using XoR between the generated nonce, the device fingerprint, and the re-ordered fingerprint value.
4) The end device encrypts the hashing output, which represents the response of the received challenge, using the NWK key and sends it to the smart gateway.
5) After receiving the authentication reply $E_{NWK}(id, nonce, H(nonce \oplus PUF \oplus RPUF)$, the smart gateway decrypts the message, checks the nonce, and compares the received response with the locally calculated one (the hashing output). If they match, that means the end device is trustable and authenticated.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2020.3032518, IEEE Internet of Things Journal

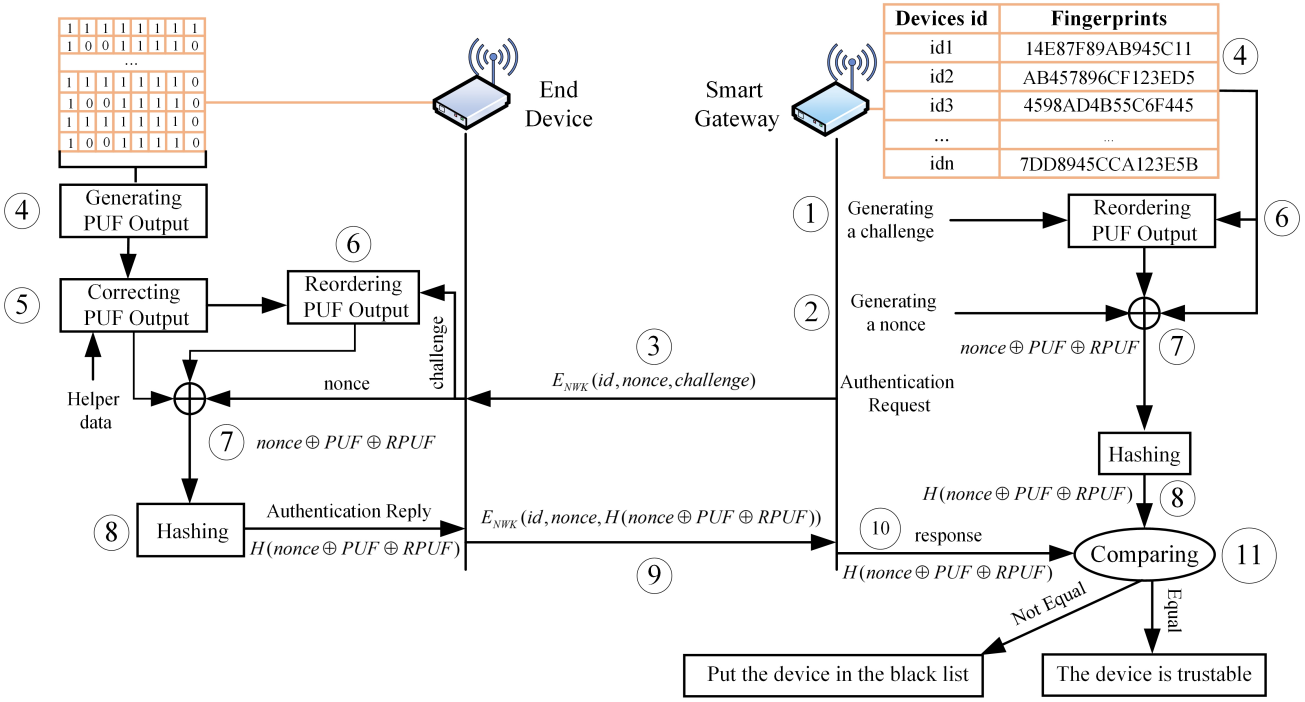


Fig. 4. Authentication Phase

Otherwise, the device will be marked as an untrustable device and added to the blacklist.

III. EXPERIMENTAL EVALUATION

A. Reading SRAM startup values

The first step in our experiment was reading the SVRCs of the end devices. The reading program was written using C language, and we read 1 KB (1*1024*8 =8192 bits) of each device 50 times. The devices in this experiment were as follows: seven Arduino Mega 2560 (Clock Speed: 16 MHz, SRAM: 8 KB), three Arduino UNO (Clock Speed: 16 MHz, SRAM: 2 KB), and one Arduino Nano (Clock Speed: 16 MHz, SRAM: 2 KB). After analyzing and comparing the readings data by writing some programs using Python programming language, we summarize some statistics and facts about the obtained results as follows:

- As shown in Fig.5, on average, the percentage of the cells that are most likely to start as one (1-skewed cells) is about 68.40% and as zero (0-skewed cells) is about 31.48%. In addition, the neutral cells (50:50 probabilities of starting as 0 or 1) represent about 0.12% of the SRAM cells. These results are very close to the ones published in [12] and [14]. For each device, the percentage of the 0-skewed cells $C0$ and of 1-skewed cells $C1$ are calculated using the eq.1 and eq.2 as follows:

$$C0 = \frac{N0}{N0 + N1 + Nn} \tag{1}$$

$$C1 = \frac{N1}{N0 + N1 + Nn} \tag{2}$$

Where $N0$ is the number of 0-skewed cells, $N1$ is the number of 1-skewed cells, and Nn is the number of neutral cells.

- As shown in Fig.6, the 1-skewed cells are a little more stable than 0-skewed cells. On average, about 88.56% of the 1-skewed cells are stable, the maximum rate is 93.53%, and the minimum rate is 83.04%. For 0-skewed cells, on average, about 84.06% of the cells are stable, the maximum rate is 88.87%, and the minimum rate is 78.11%. Before choosing the stable cells' addresses, we considered that some stable cells could be unstable under different circumstances, so we also studied the SVRCs under three conditions, as presented later in this paper. For each device, the percentage of the 100% stable 0-skewed cells $S0$ and 100% stable 1-skewed cells $S1$ are calculated using the eq.3 and eq.4 as follows:

$$S0 = \frac{NS0}{N0} \tag{3}$$

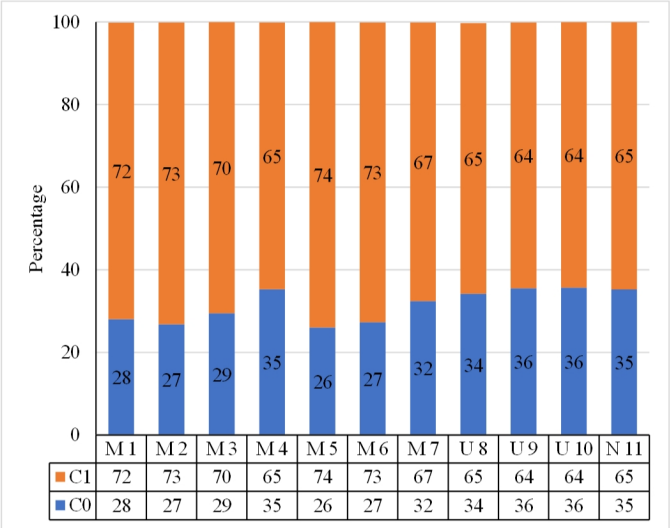
$$S1 = \frac{NS1}{N1} \tag{4}$$

Where $NS0$ is the number of the 100% stable 0-skewed cells, and $NS1$ is the number of the 100% stable 1-skewed cells.

- As shown in Fig.7, the average of the 100% stable cells extracted from the SRAM is about 87%, the maximum rate is about 92%, and the minimum rate is about 82%. Of these 100% stable cells, there were, on average, about 69% 1-skewed cells and 31% 0-skewed cells. For each device, the percentage of the 100% stable cells SC, the percentage of the 0-skewed cells in the 100% stable cells $P0$, and the percentage of the 1-skewed cells in the 100% stable cells $P1$ are calculated using the eq.5, eq.6, and eq.7 as follows:

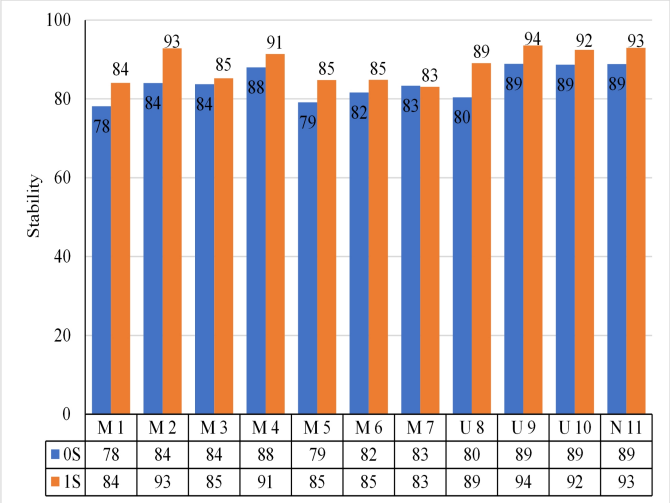
$$SC = \frac{NSx}{Ncells} \tag{5}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2020.3032518, IEEE Internet of Things Journal



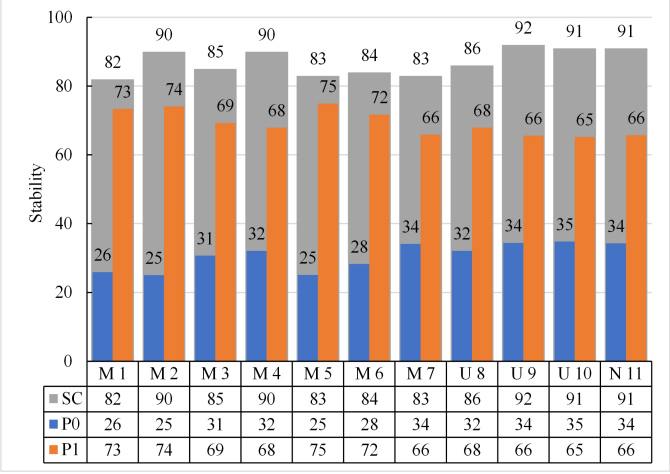
	M 1	M 2	M 3	M 4	M 5	M 6	M 7	U 8	U 9	U 10	N 11
C1	72	73	70	65	74	73	67	65	64	64	65
C0	28	27	29	35	26	27	32	34	36	36	35

Fig. 5. The Distribution of 1-skewed and 0-skewed cells in the startup values of SRAM cells (M: Arduino Mega, U: Arduino Uno, N: Arduino Nano, 0s: 0-skewed Cells Percentage, 1s: 1-skewed Cells Percentage)



	M 1	M 2	M 3	M 4	M 5	M 6	M 7	U 8	U 9	U 10	N 11
0S	78	84	84	88	79	82	83	80	89	89	89
1S	84	93	85	91	85	85	83	89	94	92	93

Fig. 6. The Stability of 0-skewed and 1-skewed Cells in the startup values of SRAM cells



	M 1	M 2	M 3	M 4	M 5	M 6	M 7	U 8	U 9	U 10	N 11
SC	82	90	85	90	83	84	83	86	92	91	91
P0	26	25	31	32	25	28	34	32	34	35	34
P1	73	74	69	68	75	72	66	68	66	65	66

Fig. 7. The Stability of the startup values of SRAM Cells with Showing Stable 0-skewed and 1-skewed Cells ratio

$$P0 = \frac{NS0}{NSx} \tag{6}$$

$$P1 = \frac{NS1}{NSx} \tag{7}$$

Where NSx is the number of 100% stable cells, whether they are 0-skewed or 1-skewed cells, and $Ncells$ is the number of tested cells.

B. The Evaluation of Temperature Effect

We increased the temperature of the microcontroller, in which SRAM is embedded, to 60°C-65°C, and then read the SVRCs. According to the obtained information, the total stability loses on average about 3.47% (4 stable cells changed their startup values when reading 100 bits), the maximum loss is 8.90% (9 stable cells changed their startup value), and the minimum loss is 0.63% (1 stable cell changed its startup value). Also, by comparing the stability of the startup values of the SRAM cells, we find out that 1-skewed cells are more vulnerable to be unstable under the high temperature comparing with 0-skewed cells. On average, the stability of 1-skewed cells loses 4.70%, whereas that of the 0-skewed cells loses 0.69%.

C. The Evaluation of Magnetic Effect

We read the SVRCs after putting a magnetic near the microcontroller in different positions. We used a magnetic with a size of 20*4 mm and magnetic field strength of 1500 Gauss. We have done this experiment under such circumstances because some sensors and actuators use magnetics, which sometimes put the microcontroller's SRAM inside a magnetic field. According to the obtained results, the total stability loses about 0.84% (1 stable cell changed its startup value) on average, the maximum loss is 5.45% (6 stable cells changed their startup value), and the minimum loss is 0.16%. By comparing the stability of 0-skewed cells and 1-skewed cells, we find out that 1-skewed cells are less vulnerable to changing their startup value than 0-skewed cells under the magnetic field. In this experiment, the stability of the 1-skewed cells loses 0.32%, whereas that of the 0-skewed cells loses 2% on average.

D. The Evaluation of Previous Written Value Effect

SRAM-PUF was tested by writing zeros and ones to all of the SRAM cells used as PUF. After doing that, we turned the device OFF and then ON to read the SVRCs. The purpose of this experiment is to figure out whether the previous states of the cells affect the stability of their startup values. For each device, we read the SVRCs five times after writing zeros and five times after writing ones. Then we analyzed the results and compared them with the reading values of the devices in normal cases. We found out that the total stability loses about 0.99% (1 stable cell changed its startup value) on average, the maximum loss is about 4.98% (5 stable cells changed its startup value), and the minimum loss is about 0.1%. Besides, by comparing the stability of 0-skewed cells and 1-skewed cells, we can figure out that 1-skewed cells (loss 0.58%) are less vulnerable to changing their startup values than 0-skewed (loss 1.86%) after writing different values on the cells.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2020.3032518, IEEE Internet of Things Journal

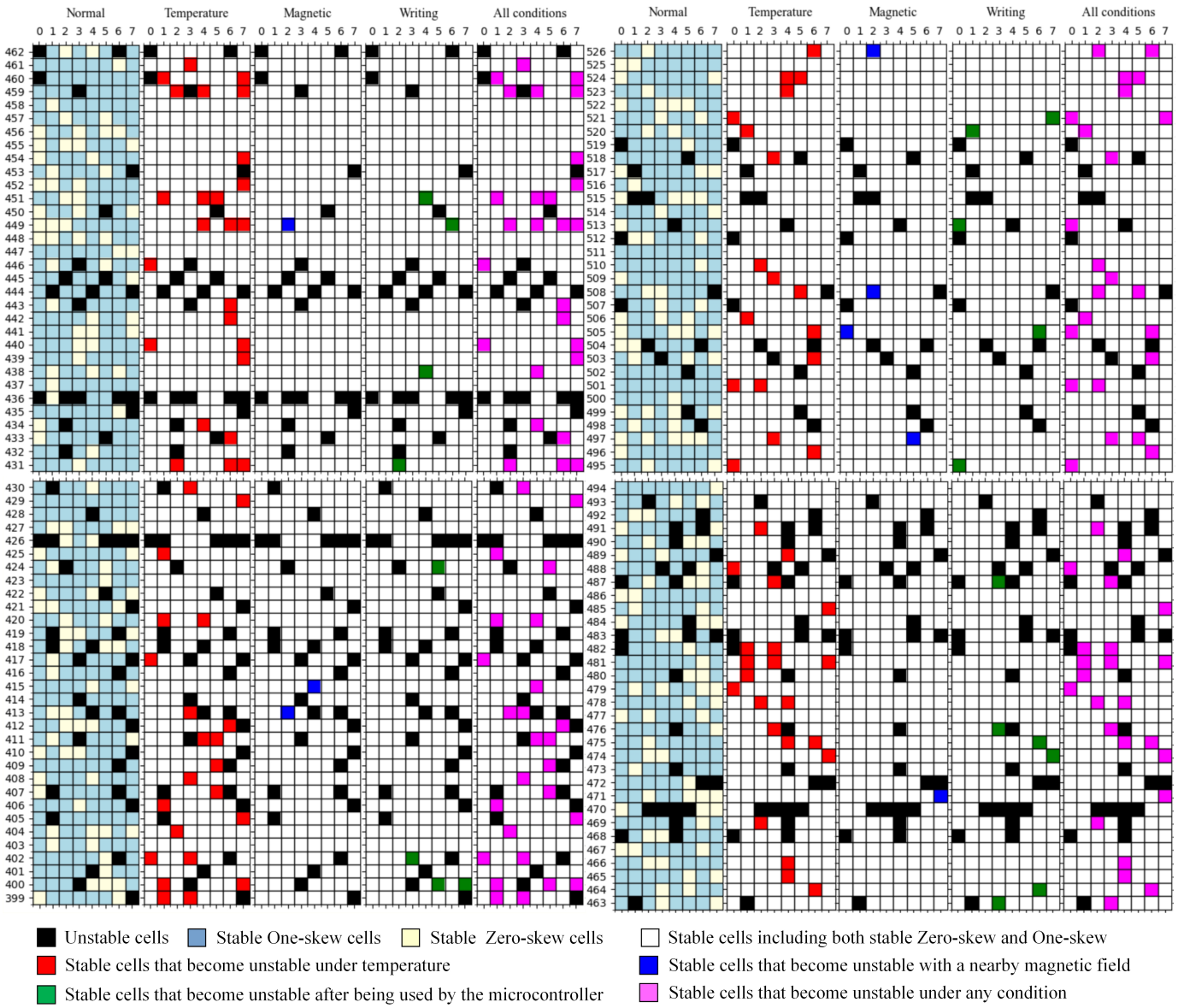


Fig. 8. The Stability of SRAM Cells under different circumstances (Temperature, Magnetic, and Writing on the cells) for the device 2 (Arduino Mega)

E. Summary and discussion of the environmental effects

According to the results in this experiment, 1-skewed cells are more likely to be unstable under high temperature but more stable against the nearby magnetic field effect and the effect of the previously written value compared with 0-skewed cells. Fig.8 shows the stable cells of device 2 (Arduino Mega) and how they changed their startup values under different conditions. The "Normal" part represents the SVRCs under the room temperature of 25°C. "Temperature," "Magnetic," and "Writing" parts represent the SVRCs under the three tested conditions, respectively. The last part "All Conditions," shows all the stable cells that turned into unstable cells under one or more of the tested conditions.

As shown in TABLE I, there are 0.003% of the cells changed under all the conditions, and 0.013%, 0.05%, 0.013% changed under the temperature and magnetic, temperature and writing, and magnetic and writing, respectively. As a result, the cells that lose stability under one condition are mostly different from the cells which lose stability under other conditions. The accumulated bit-error from all the tests in case they are totally independent is 9% + 6% + 5% = 21%. In fact, having 21 bit-error in one reading is very unlikely to happen. The 21 bit-error happened in different readings. However, the error-correcting coding (ECC) algorithm was chosen according to the worst scenario and can fix up to 22 bit-error for a 100-bit fingerprint, which will make the extracted fingerprint %100 stable even under different environmental conditions.

IV. Results and Discussion

The authentication request message consists of a nonce with a length of 4 bytes and a challenge with a length of 100 bytes. This challenge consists of the order of the participating memory cells. It could be the addresses of PUF cells, but that requires a challenge of 200 bytes since the SRAM cells'

TABLE I
UNSTABLE CELLS UNDER ENVIRONMENTAL CONDITIONS

Devices	TMW	TM	TW	MW	T	M	W	Stable
M1	4	4	15	5	85	11	63	6548
M2	0	10	98	8	539	52	39	6648
M3	0	4	6	2	110	5	22	6787
M4	2	6	15	1	306	11	8	7042
M5	0	1	4	0	106	19	5	6675
M6	0	2	6	1	88	16	1	6751
M7	1	1	1	2	40	16	4	6734
U8	0	7	5	2	267	13	25	7173
U9	1	2	12	9	383	16	28	7067
U10	1	4	2	13	265	21	18	7132
N11	0	7	5	2	267	13	25	7173
MAX	4	10	98	13	539	52	63	7173
MIN	0	1	1	0	40	5	1	6548
AVR	1	4	15	4	223	18	22	6885

T: Temperature, M: Magnetic, W: After writing 0 or 1 to the cell, TM: changed in both Temperature and Magnetic, TW: changed in both Temperature and writing, MW: changed in both Magnetic and writing, TMW: changed in Temperature , Magnetic, and writing. (MEMORY SIZE=8192 BITS)

address length is 2 bytes (enough to address up to 64 KB memory cells). Besides, sending the addresses of SRAM-PUF cells is considered as a security threat.

Authentication with a 100-bit fingerprint was firstly tested using Arduino Mega. On the end device side, the process took about 34ms, including SRAM cells reading, BCH decoding, and MD5 Hashing. The written program's global variables occupied about 1.7 KB of SRAM, and the program size occupied about 7 KB of the flash memory, including 148 bits of the helping data generated by the BCH encoding, the source code for BCH decoding, and the hashing function MD5. BCH ran as (255,107,45), which means that the fingerprint's length can be up to 107 bits, the final block size (data + helper data) is 255 bits, and this code can correct up to 22 bits. However, using the same authentication method on Arduino Uno and Nano, which have an SRAM of 2KB, is not feasible, so we had to reduce the fingerprint's length to 50 bits and run the BCH as (127, 57, 23). That means the length of the fingerprint size can be up to 57 bits, the final block size (data + helper data) is 127 bits, and the BCH code can correct up to 11 bits. As a result, to generate a fingerprint using our scheme, the device should have an SRAM memory with a size more than 2KB.

Choosing whether to store the helper data on the end device or the server depends on balancing communication and storage. In general, when a strong PUF is used, the helper data are stored on the server (the authenticator). That is because each response could be noisy and need helper data to be corrected, which makes the size of the helper data significant. Besides, the end devices could be resources-constrained and do not have enough storage capacity to store all these data. Using weak PUF like SRAM-PUF, the helper data can be either stored on the device or the server. That is because the number of CRPs is small, and so is the size of the helper data. In this paper, the used PUF is weak. Even though the number of CRPs is large, there is just one SRAM-PUF output required to generate all the responses. Accordingly, if the fingerprint's length is 100 bits, then the size of the helper data needed to correct 22 bit-error is 148 bits. Therefore, if there is enough space on the end device, the helper data should be stored there to eliminate the communication overhead caused by sending the data every time the authentication process is carried out. If there is no storage capacity, the helper data will be stored on the server and sent to the end device at the beginning of every authentication process.

If the authentication process happens just once each time the device boots up and joins the system, the end device can hide its fingerprint by storing a random value of 100 bits on the SRAM-PUF cells. Doing that will hide the PUF output, but it will prevent the device from doing other authentication operations during the run time because the SRAM-PUF output can only be obtained when the device boots up. To address this issue, we can keep the PUF output stored in an array inside the SRAM, which occupies 13 bytes while hiding the startup values of the SRAM-PUF cells. There was a proposed structure of the SRAM, which can reset individual cells anytime [30]. That includes adding four transistors to each SRAM cell, and thus, any cell can be reset anytime without the need to restart the device in order to read the SRAM-PUF output. However, this solution requires changing the SRAM structure, which is out of this paper's scope.

From a security perspective, our proposed scheme is robust against the attacks intended to authenticate a fake end device which does not belong to the system. Some attacks are discussed below:

- The Man in the Middle Attack: in this attack, an adversary can listen and capture authentication request/reply messages to be used later in the replay attacks. If a previous authentication request got repeated, i.e., the sever chose a challenge and a nonce that were used before, the adversary can successfully authenticate a fake entity by sending the captured response. However, for this attack to carry out against our scheme, a challenge of 100 numbers and a nonce of 4 bytes chosen randomly need to be repeated, which is very unlikely to happen.
- Side-Channel Attack: in this attack, the intruder can physically access the end device and read the PUF output. However, this attack is almost impossible to succeed with the SRAM-PUF. That is because SRAM is embedded in the microcontroller package and has no connection to the outside world. Thus, the SRAM cannot be read without corrupting the microcontroller.
- Extracting hashing tables: the hashing tables are usually stored in the NVM, making them vulnerable to the side-channel attack. Extracting the hashing tables will compromise the whole security system and may help in exposing the PUF output. Therefore, the hashing tables in the NVM need protection by encrypting them using a hardware key derived from the SRAM-PUF [12].
- Brute force attack: when the 100-bit fingerprint is extracted directly from the SVRCs, it will have, theoretically, 68 ones and 32 zeros inherited from the 0:1 ratio

TABLE II
COMPARISON BETWEEN THE PUF-BASED SCHEMES

Scheme	CO	Neq	HDS	NCRPs
Wallrabenstein, J.R. [18]	Medium	Yes	Small	Small
Braeken, A. [19]	High	Yes	Big	Big
Che, Wenjie, et al. [20]	Small	Yes	Big	Big
Gope et al. [21]	Medium	Yes	Big	Big
Obfuscated protocols [22]	Small	Yes	Medium	Medium
Our scheme	Small	No	Small	Big

CO: Computation overhead for resource-constrained end devices, Neq: requires installing a new equipment, HDS: Helper Data size, NCRPs: Number of CRPs.

found in SVRCs. Therefore, in this paper, we choose the participating SRAM addresses to ensures that 50% of the fingerprint are 0s and the other 50% are 1s. That will make it more difficult for attackers to guess all the possible compositions (Brute force attack). When we generate a 100-bit key, there are $5.0446 * 10^{28}$ compositions for 50:50 ratio of 0s and 1s, and $4.576 * 10^{25}$ compositions for 32:68 ratio of 0s and 1s.

Finally, Our scheme can provide unique fingerprints to the end devices without adding any new equipment in addition to a significant number of CRPs with a small size of data helper. Besides, the computation overhead of the authentication process is comparatively small and can be handled by the resource-constrained devices. TABLE II presents a comparison between our scheme and the previously proposed schemes.

V. CONCLUSION

Remote sensing and control are an essential part of our digital world. Accessing resources that are far away is becoming a common practice now. Therefore, the entity authentication of remote devices becomes a hot topic while building a trust system. Running the traditional authentication algorithm on the resource-constrained devices faces some challenges related to computing power and memory storage. Therefore, this paper proposed a lightweight SRAM-PUF based authentication scheme to ensure the end devices' entity. After analyzing and testing the startup values of SRAM cells under different conditions, a stable fingerprint was extracted and then corrected using BCH. The experiment in this paper shows that the proposed scheme can be deployed in the IoT end devices and perform the entity authentication efficiently. Besides, its requirements can be easily satisfied by the resource-constrained devices. In future work, there will be more tests to run on the SRAM cells and more in-depth research to find a module to predict which cells are more likely to be unstable under different conditions.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China (61872038), Scientific and Technological Innovation Foundation of Shunde Graduate School, USTB (BK19CF010), UK Royal Society-Newton Mobility Grant (No.IEC\NSFC\170067), and in part by the Fundamental Research Funds for the Central Universities under Grant FRF-GF-19-020B.

REFERENCES

[1] J. Ding, M. Nemati, C. Ranaweera, and J. Choi, "IoT Connectivity Technologies and Applications: A Survey," *IEEE Access*, vol. 8, pp. 67 646–67 673, Feb 2020.

[2] Y. Chen, W. Sun, N. Zhang, Q. Zheng, W. Lou, and Y. T. Hou, "Towards Efficient Fine-Grained Access Control and Trustworthy Data Processing for Remote Monitoring Services in IoT," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 7, pp. 1830–1842, Dec 2019.

[3] B. Martinez, M. Montón, I. Vilajosana, and J. D. Prades, "The power of models: Modeling power consumption for iot devices," *IEEE Sensors Journal*, vol. 15, no. 10, pp. 5777–5789, 2015.

[4] C. Boyd, A. Mathuria, and D. Stebila, *Authentication and Key Transport Using Public Key Cryptography*. Berlin, HeidelbHolcomb2009erg: Springer Berlin Heidelberg, 2020, pp. 135–164.

[5] S. L. Keoh, S. S. Kumar, and H. Tschofenig, "Securing the internet of things: A standardization perspective," *IEEE Internet of Things Journal*, vol. 1, no. 3, pp. 265–275, 2014.

[6] Q. Wang and G. Qu, "A Silicon PUF Based Entropy Pump," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 3, pp. 402–414, Nov 2019.

[7] A. Shamir, "Identity-Based Cryptosystems and Signature Schemes," in *Advances in Cryptology*, G. R. Blakley and D. Chaum, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1985, pp. 47–53.

[8] C. Peikert and Others, "A decade of lattice cryptography," *Foundations and Trends®in Theoretical Computer Science*, vol. 10, no. 4, pp. 283–424, Mar 2016.

[9] S. Gorbunov, V. Vaikuntanathan, and H. Wee, "Attribute-Based Encryption for Circuits," *Journal of the ACM (JACM)*, vol. 62, no. 6, pp. 1–33, Dec 2015.

[10] D. He, M. Ma, S. Zeadally, N. Kumar, and K. Liang, "Certificateless public key authenticated encryption with keyword search for industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 8, pp. 3618–3627, 2018.

[11] D. Boneh, C. Gentry, S. Gorbunov, S. Halevi, V. Nikolaenko, G. Segev, V. Vaikuntanathan, and D. Vinayagamurthy, "Fully Key-Homomorphic Encryption, Arithmetic Circuit ABE and Compact Garbled Circuits," in *Advances in Cryptology – EUROCRYPT 2014*, P. Q. Nguyen and E. Oswald, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 533–556.

[12] F. Farha, H. Ning, H. Liu, L. Yang, and L. Chen, "Physical unclonable functions based secret keys scheme for securing big data infrastructure communication," *Information Sciences*, vol. 503, Nov 2019.

[13] M. N. Aman, K. C. Chua, and B. Sikdar, "Mutual authentication in iot systems using physical unclonable functions," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1327–1340, 2017.

[14] D. Holcomb, W. Burleson, and K. Fu, "Power-Up SRAM State as an Identifying Fingerprint and Source of True Random Numbers," *IEEE Transactions on Computers*, vol. 58, no. 9, pp. 1198–1210, Nov 2009. [Online]. Available: http://ieeexplore.ieee.org/document/4674345/

[15] H. Ning, F. Farha, A. Ullah, and L. Mao, "Physical unclonable function: architectures, applications and challenges for dependable security," *IET Circuits, Devices & Systems*, Feb 2020.

[16] J. Guajardo, B. Škorić, P. Tuyls, S. S. Kumar, T. Bel, A. H. M. Blom, and G.-J. Schrijen, "Anti-counterfeiting, key distribution, and key storage in an ambient world via physical unclonable functions," *Information Systems Frontiers*, vol. 11, no. 1, pp. 19–41, Mar 2009.

[17] J. Delvaux, D. Gu, D. Schellekens, and I. Verbauwhede, "Helper data algorithms for puf-based key generation: Overview and analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 6, pp. 889–902, 2015.

[18] J. R. Wallrabenstein, "Practical and Secure IoT Device Authentication Using Physical Unclonable Functions," in *2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud)*, Vienna, Austria, 2016, pp. 99–106.

[19] A. Braeken, "PUF based authentication protocol for IoT," *Symmetry*, vol. 10, no. 8, pp. 352–367, Jul 2018.

[20] W. Che, M. Martin, G. Pocklassery, V. Kajuluri, F. Saqib, and J. Plusquellic, "A Privacy-Preserving, Mutual PUF-Based Authentication Protocol," *Cryptography*, vol. 1, no. 1, p. 3, Nov 2016.

[21] P. Gope, A. K. Das, N. Kumar, and Y. Cheng, "Lightweight and physically secure anonymous mutual authentication protocol for real-time data access in industrial wireless sensor networks," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 9, pp. 4957–4968, 2019.

[22] Y. Gao, G. Li, H. Ma, S. F. Al-Sarawi, O. Kavehei, D. Abbott, and D. C. Ranasinghe, "Obfuscated challenge-response: A secure lightweight authentication mechanism for puf-based pervasive devices," in *2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, 2016, pp. 1–6.

[23] F. Tehranipoor, N. Karimian, W. Yan, and J. A. Chandy, "DRAM-based intrinsic physically unclonable functions for system-level security and authentication," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 3, pp. 1085–1097, Sep 2017.

[24] P. Prabhu, A. Akel, L. M. Grupp, W.-K. S. Yu, G. E. Suh, E. Kan, and S. Swanson, "Extracting Device Fingerprints from Flash Memory by Exploiting Physical Variations," in *Trust and Trustworthy Computing*, J. M. McCune, B. Balacheff, A. Perrig, A.-R. Sadeghi, A. Sasse, and Y. Beres, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 188–201.

[25] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls, "FPGA intrinsic PUFs and their use for IP protection," in *Proc. 9th International Workshop on Cryptographic Hardware and Embedded Systems - CHES 2007*, Vienna, 2007, pp. 63–80.

[26] D. E. Holcomb, W. P. Burleson, K. Fu, and Others, "Initial SRAM state as a fingerprint and source of true random numbers for RFID tags," in *Proceedings of the Conference on RFID Security*, vol. 7, no. 2, 2007, p. 1.

[27] R. Maes and V. van der Leest, "Countering the effects of silicon aging on sram pufs," in *2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2014, pp. 148–153.

[28] R. Wang, G. Selimis, R. Maes, and S. Goossens, "Long-term continuous assessment of sram puf and source of random numbers," in *2020 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2020, pp. 7–12.

[29] Y. Xiao, Y. Jia, C. Liu, X. Cheng, J. Yu, and W. Lv, "Edge Computing Security: State of the Art and Challenges," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1608–1631, Jun 2019.

[30] Y. Su, J. Holleman, and B. P. Otis, "A Digital 1.6 pJ/bit Chip Identification Circuit Using Process Variations," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 1, pp. 69–77, Jan 2008.

Fadi Farha received his BS from the faculty of Informatics Engineering, Aleppo University, Syria. He did his MS degree and currently pursuing toward a Ph.D. degree in the School of Computer and Communication Engineering, University of Science and Technology Beijing, China. His current research interests include Physical Unclonable Function (PUF), Security Solutions, ZigBee, Computer Architecture, and Hardware Security.

Huansheng Ning (SM'13) received his B.S. degree from Anhui University in 1996 and his Ph.D. degree from Beihang University in 2001. He is currently a Professor and Vice Dean with the School of Computer and Communication Engineering, University of Science and Technology Beijing, China, and the founder and principal at Cybermatics and Cyberspace International Science and Technology Cooperation Base. He has authored 6 books and over 180 papers in journals and at international conferences/workshops. He has been the Associate Editor of IEEE Systems Journal, the associate editor (2014-2018), area editor (2020-) and the Steering Committee Member of IEEE Internet of Things Journal (2018-). He is the host of the 2013 IEEE Cybrmatics Congress and 2015 IEEE Smart World Congress. His awards include the IEEE Computer Society Meritorious Service Award and the IEEE Computer Society Golden Core Member Award. His current research interests include Internet of Things, Cyber Physical Social Systems, Cyberspace Data and Intelligence.

Karim Ali received his Bachelor's degree from the Department of Computer Engineering and Communications, Lebanese International University, Lebanon. He is currently pursuing toward Master's degree in Computer Science and Technology, University of Science and Technology Beijing, China. His research focuses mainly on the vulnerabilities of machine learning and adversarial attacks against deep learning.

Liming Chen received the B.E. and M.E. degrees from the Beijing Institute of Technology, Beijing, China,and the Ph.D. degree in artificial intelligence from DeMontfort University, Leicester, U.K. He is currently a Professor of Data Analytics, Research Director for the School of Computing, and lead the Cognitive Analytics Research Lab (CARL) at Jordanstown campus in Ulster University (UU). His research interests include pattern recognition, intelligent systems, smart environment, and assisted living.

Christopher Nugent is the Head of School of Computing and holds the position of Professor of Biomedical Engineering. He is based within the School of Computing and Mathematics at Ulster University. He received a Bachelor of Engineering in Electronic Systems and DPhil in Biomedical Engineering both from Ulster University. Chris joined the University as a Research Fellow in 1999 and was appointed as Lecture in Computer Science in 2000. Following this he held positions of Senior Lecture and Reader within the Faculty of Computing and Engineering before his appointment as Professor of Biomedical Engineering in 2008. In 2016 he was awarded the Senior Distinguished Research Fellowship from Ulster University, His research within biomedical engineering addresses the themes of the development and evaluation of technologies to support ambient assisted living. Specifically, this has involved research in the topics of mobile based reminding solutions, activity recognition and behaviour modelling and more recently technology adoption modelling. He has published extensively in these areas with papers spanning theoretical, clinical and biomedical engineering domains. He has been a grant holder of Research Projects funded by National, European and International funding bodies. Chris is the Group Leader of the Smart Environments Research Group and also the co-PI of the Connected Health Innovation Centre at Ulster University.